日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２０００年　５月３１日

出願番号
Application Number: 特願２０００－１６３７９２

出願人
Applicant(s): 株式会社東芝

２００１年　５月１１日

特許庁長官
Commissioner,
Japan Patent Office

及川耕造

Docket No. 209257US2SRD

#2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Katsuhiko UEKI, et al. GAU:

SERIAL NO: New Application EXAMINER:

FILED: Herewith

FOR: COMPUTER SYSTEM AND METHOD FOR AIDING LOG BASE DEBUGGING

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number, filed, is claimed pursuant to the provisions of **35 U.S.C. §120**.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.
- ■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
JAPAN	2000-163792	May 31, 2000

Certified copies of the corresponding Convention Application(s)

- ■ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number .
  Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
  (B) Application Serial No.(s)
  - ☐ are submitted herewith
  - ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913
C. Irvin McClelland
Registration Number 21,124

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

【書類名】　特許願

【整理番号】　A000002886

【提出日】　平成12年 5月31日

【あて先】　特許庁長官　殿

【国際特許分類】　G06F 17/00

【発明の名称】　ログ比較デバッグ支援装置

【請求項の数】　5

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　植木　克彦

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　田村　文隆

【発明者】

【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内

【氏名】　岡本　渉

【特許出願人】

【識別番号】　000003078

【氏名又は名称】　株式会社　東芝

【代理人】

【識別番号】　100058479

【弁理士】

【氏名又は名称】　鈴江　武彦

【電話番号】　03-3502-3181

【選任した代理人】

【識別番号】　100084618

【弁理士】

【氏名又は名称】　村松　貞男

【選任した代理人】

【識別番号】　100068814

【弁理士】

【氏名又は名称】　坪井　淳

【選任した代理人】

【識別番号】　100092196

【弁理士】

【氏名又は名称】　橋本　良郎

【選任した代理人】

【識別番号】　100091351

【弁理士】

【氏名又は名称】　河野　哲

【選任した代理人】

【識別番号】　100088683

【弁理士】

【氏名又は名称】　中村　誠

【選任した代理人】

【識別番号】　100070437

【弁理士】

【氏名又は名称】　河井　将次

【手数料の表示】

【予納台帳番号】　011567

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　　要

【書類名】　明細書

【発明の名称】　ログ比較デバッグ支援装置

【特許請求の範囲】

【請求項１】　対象プログラムの実行によって発生した一連のイベントが記録されたログを入力し、所定の条件に従って該入力ログから複数の部分ログを生成する部分ログ生成手段と、

前記条件に対応する正規化規準としての主ログに基づき、前記部分ログを正規化する正規化ログ生成手段と、

前記正規化ログ生成手段により生成された正規化ログのそれぞれについて、前記イベントの発生及び不発生の特徴度合いを示す特徴値を、他の部分ログの正規化ログに基づいて算出する特徴値算出手段と、

所定の部分ログと他の部分ログとの組み合わせにおいて、該部分ログ同士の類似度を前記特徴値に基づく所定の演算により算出する類似度算出手段と、

を具備することを特徴とするログ比較デバッグ支援装置。

【請求項２】　特定の部分ログを指定する部分ログ指定手段と、

前記部分ログ指定手段により指定された部分ログとの類似度が高い部分ログを、前記類似度算出手段により算出された類似度に基づき選択する部分ログ選択手段と、

を具備することを特徴とする請求項１に記載のログ比較デバッグ支援装置。

【請求項３】　前記条件として、前記部分ログの先頭及び末尾のイベントを指定し、及び該イベント間に挟まれたイベント列の少なくとも一部を抽出するための抽出規則を指定する条件指定手段と、

前記部分ログ生成手段により生成された複数の部分ログを、不要な重複イベントを除外しながら結合することにより前記主ログを生成する主ログ生成手段と、

を具備することを特徴とする請求項１又は２のいずれかに記載のログ比較デバッグ支援装置。

【請求項４】　前記条件としてソースプログラムの特定関数を指定し、及び該特定関数内における関数の呼び出し及びループを含む特定構文に係る展開パラメータを指定する条件指定手段と、

前記対象プログラムの入力ログに対応するソースプログラムを入力し、該ソースプログラムにおける前記特定関数に相当する記述を、前記展開パラメータに従って展開し、該展開結果を前記主ログとして生成する主ログ生成手段と、

を具備することを特徴とする請求項１又は２のいずれかに記載のログ比較デバッグ支援装置。

【請求項５】　前記展開パラメータは、関数の再帰的呼び出し及びループ構文の最小展開数を指定可能に構成されることを特徴とする請求項４に記載のログ比較デバッグ支援装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、対象プログラムの実行によって発生した一連のイベントが記録（トレース）されたログを利用するデバッグに好適なログ比較デバッグ支援装置に関する。

【０００２】

【従来の技術】

プログラム中に含まれる誤り（バグ）を正す作業（「デバッグ」という）をプログラマー等が行なう際には、これを支援するデバッガ（デバッグ支援装置）が利用される。デバッガは、デバッグ作業者からの指示に従って対象プログラムを起動し、その実行を制御する。いわゆるデバッグ実行の過程において、デバッグに有益な各種情報を表示することができる。

【０００３】

デバッグには幾つかの手法が知られているが、最近では、プログラムの動作確認やデバッグ作業のために動作ログを使用する場合が増えてきている。例えば、システムコール発行の履歴をＯＳにより記録し、メモリアクセスの履歴をハードウェア又はエミュレータにより記録し、これらの集合としてのイベント情報（ログ）を、専用のビューアを用いて表示してデバッグの一助とする、ログベースのデバッグ手法が普及してきている。

【０００４】

ログを用いてデバッグを行う場合、一つのログを詳細に調査してバグの原因を突き止める手法のほかに、正常動作した場合のログと異常動作した場合のログとを比較して相違部分を探し出し、かかる相違部分を集中的に調査することでバグの原因を突き止める手法がある。

【０００５】

ところで、このようなログ比較に基づくデバッグでは、不適当なログを選択して比較を行なってしまうと相違部分が多く見つかり過ぎて調査範囲が拡大し、デバッグ効率が著しく低下してしまう。そこで、できるだけ似た挙動を示すログをデバッグ作業者が選択し、比較に用いるようにしている。

【０００６】

本来ならば、プログラムの動作の意味を考慮して類似ログを選択すべきだが、従来の手法は、ログ同士の構成内容を単に比較して最も似ているログを選択するだけのものであり、このためループの繰り返し回数が最も近いものが選択される場合があるなど、必ずしも効率的ではなかった。

【０００７】

【発明が解決しようとする課題】

本発明は上記事情を考慮してなされたものであり、その目的は、ログ比較に有益な部分ログを得ることができ、ログベースのデバッグ作業の効率向上に寄与するログ比較デバッグ支援装置を提供することにある。

【０００８】

【課題を解決するための手段】

上記課題を解決し目的を達成するために、本発明は次のように構成されている。

【０００９】

（１）本発明のログ比較デバッグ支援装置は、対象プログラムの実行によって発生した一連のイベントが記録されたログを入力し、所定の条件に従って該入力ログから複数の部分ログを生成する部分ログ生成手段と、前記条件に対応する正規化規準としての主ログに基づき、前記部分ログを正規化する正規化ログ生成手段と、前記正規化ログ生成手段により生成された正規化ログのそれぞれについて、

前記イベントの発生及び不発生の特徴度合いを示す特徴値を、他の部分ログの正規化ログに基づいて算出する特徴値算出手段と、所定の部分ログと他の部分ログとの組み合わせにおいて、該部分ログ同士の類似度を前記特徴値に基づく所定の演算により算出する類似度算出手段と、を具備することを特徴とするログ比較デバッグ支援装置である。

【００１０】

（２）本発明のログ比較デバッグ支援装置は、上記（１）に記載の装置であって、かつ特定の部分ログを指定する部分ログ指定手段と、前記部分ログ指定手段により指定された部分ログとの類似度が高い部分ログを、前記類似度算出手段により算出された類似度に基づき選択する部分ログ選択手段と、を具備することを特徴とするログ比較デバッグ支援装置である。

【００１１】

（３）本発明のログ比較デバッグ支援装置は、上記（１）又は（２）のいずれかに記載の装置であって、かつ前記条件として、前記部分ログの先頭及び末尾のイベントを指定し、及び該イベント間に挟まれたイベント列の少なくとも一部を抽出するための抽出規則を指定する条件指定手段と、前記部分ログ生成手段により生成された複数の部分ログを、不要な重複イベントを除外しながら結合することにより前記主ログを生成する主ログ生成手段と、を具備することを特徴とするログ比較デバッグ支援装置である。

【００１２】

（４）本発明のログ比較デバッグ支援装置は、上記（１）又は（２）のいずれかに記載の装置であって、かつ前記条件としてソースプログラムの特定関数を指定し、及び該特定関数内における関数の呼び出し及びループを含む特定構文に係る展開パラメータを指定する条件指定手段と、前記対象プログラムの入力ログに対応するソースプログラムを入力し、該ソースプログラムにおける前記特定関数に相当する記述を、前記展開パラメータに従って展開し、該展開結果を前記主ログとして生成する主ログ生成手段と、を具備することを特徴とするログ比較デバッグ支援装置である。

【００１３】

（５）本発明のログ比較デバッグ支援装置は、上記（４）に記載の装置であって、かつ前記展開パラメータは、関数の再帰的呼び出し及びループ構文の最小展開数を指定可能に構成されることを特徴とするログ比較デバッグ支援装置である。

【００１４】

【発明の実施の形態】

以下、図面を参照しながら本発明の実施形態を説明する。

（第１実施形態）

図１は、本発明の第１実施形態に係るログ比較デバッグ支援装置の概略構成を示すブロック図である。図１に示すように本装置は、条件指定部１、部分ログ生成部２、主ログ生成部４、正規化ログ生成部５、ログ特徴値算出部６、ログ類似度算出部８、ログ選択部９、ログ表示部１０、部分ログ指定部１１から構成されている。

準備された動作ログＬを本実施形態のログ比較デバッグ支援装置に与え、およびデバッグ作業者１２が条件指定部１及び部分ログ指定部１１を介してログ比較に係る条件等を指定すると、ログの類似情報がログ表示部１０に表示される構成である。

【００１５】

動作ログＬは、図示しない対象プログラムの実行によって発生した一連のイベントが記録されて成る。対象プログラムの種別、数量等は任意であり、実行環境も任意である。例えば対象プログラムの一部がシミュレータにより置き換えられていたり、対象プログラムが制御するハードウェアの一部又は全部がエミュレータにより置き換えられていてもよい。

動作ログＬを準備するための対象プログラムの実行は、図示しないデバッガによる。イベントの収集・記録処理は、デバッガに組み込まれ、或いはデバッガとは別構成で協調動作する「トレーサ」により行なわれる。

本発明でいう「イベント（事象）」は、デバッグ環境全体で予め定義されるものである。かかるイベントの定義に沿って、トレーサは対象プログラムの実行動作を追跡し、デバッガによる対象プログラムのデバッグ実行動作単位（シーケンス）ごとに、発生（成立）した一連のイベントを動作ログＬに記録する。

【００１６】

図２は、本実施形態に係るログ比較デバッグ支援装置の大まかな処理の流れを示す図である。以下、第１実施形態では、対象プログラムの実行によって発生した一連のイベントが記録された動作ログＬの一例として、以下３つの動作ログ１～３が与えられた場合を考える。

動作ログ１：ａｂｃｇｈ

動作ログ２：ａｃｆｇｈ

動作ログ３：ａｂｃｄｅｇｈａｃｇｈ

ａ～ｈはそれぞれ、ログを構成する識別可能なイベントであり、その並びがイベントの発生順序を表すものである。

【００１７】

先ず、ステップＳ０において、デバッグ作業者（ユーザ）１２は、ログ比較条件として、部分ログの先頭及び末尾のイベントを指定し、及び該イベント間に挟まれたイベント列の少なくとも一部を抽出するための抽出規則を指定する（条件指定部１）。

具体的には例えば、部分ログの先頭及び末尾のイベントが、それぞれイベントａからイベントｈまでと指定され、また、イベントａとイベントｈとの間に挟まれたイベント列の抽出規則として「全てのイベントを抽出する」という規則がログ比較条件として指定されたとする。ここで、「全てのイベント」とは、部分ログとして切り出すイベントの種類を限定しない、ということを意味する。

【００１８】

ステップＳ１において、部分ログ生成部２は、動作ログＬを入力し、ステップＳ０において指定されたログ比較条件に従って、動作ログＬから複数の部分ログを生成する。ここでは、動作ログ１～３から、先頭及び末尾のイベントであるイベントａ及びｈにより挟まれたイベント列を部分ログとして切り出す。

【００１９】

結果として次に示すように、上記動作ログ１から部分ログＡ、動作ログ２から部分ログＢ、動作ログ３から部分ログＣおよび部分ログＤの計４つのログが切り出される。

部分ログＡ：（ａ）ｂｃｇ（ｈ）

部分ログＢ：（ａ）ｃｆｇ（ｈ）

部分ログＣ：（ａ）ｂｃｄｅｇ（ｈ）

部分ログＤ：（ａ）ｃｇ（ｈ）

なお、部分ログＡ～Ｄのそれぞれにおいて、先頭及び末尾のイベント（すなわちイベントａ及びｈ）は、各部分ログ間で共通であるから各々の部分ログから削除しておく。

【００２０】

次にステップＳ２において、主ログ生成部４は部分ログ生成部２により生成された部分ログＡ～Ｄを、所定の結合アルゴリズムに従い、不要な重複イベントを除外しながら結合することにより主ログを生成する。なお主ログの生成を、当該第１実施形態のように部分ログ（動作ログＬ）に基づいて生成するのではなく、後述する第２実施形態のように、動作ログＬとは異質のデータであるソース（原始）プログラムを展開処理することにより生成してもよい。

【００２１】

第１実施形態では、次に示すように部分ログＡ～Ｄの結合処理がなされ、その結合結果を主ログとする。

Ａ：　　ｂｃｇ

Ａ＋Ｂ：　　ｂｃｆｇ

（Ａ＋Ｂ）＋Ｃ：　　ｂｃｆｄｅｇ

（（Ａ＋Ｂ）＋Ｃ）＋Ｄ：　ｂｃｆｄｅｇ

生成された主ログは、条件指定部１において指定されたログ比較条件に対応する正規化規準として用いられる。

【００２２】

次にステップＳ３において、正規化ログ生成部５は、ステップＳ２において生成された主ログに基づき、部分ログＡ～Ｄを正規化する。すなわち、正規化ログ生成部５は、部分ログＡ～Ｄのそれぞれに、主ログ生成部３から得られた主ログを対応させ、主ログの構成要素イベントが当該部分ログに存在する場合には１、存在しない場合には０が設定されたビット列を生成する。これを正規化ログとす

る。

.当該第１実施形態において、部分ログＡ～Ｄそれぞれの正規化ログは、次のようになる。

主ログ：　　ｂｃｆｄｅｇ

正規化ログＡ：（１１００００１）

正規化ログＢ：（０１１００１）

正規化ログＣ：（１１０１１１）

正規化ログＤ：（０１００００１）

次にステップＳ４において、特徴値算出部６は正規化ログ生成部５により生成された正規化ログＡ～Ｄのそれぞれについて、イベントの発生及び不発生の特徴度合いを示す特徴値を、他の部分ログの正規化ログに基づいて算出する。

より詳しくは、各正規化ログの各イベントの正規化値について、

（１）自分が１の値を持つ場合（イベント発生）には、他の正規化ログにおいて該当イベントが０の値を持つ（イベント不発生）ログを集計し、その値をログ特徴として設定する。このログ特徴の値が高ければ、当該イベントの発生（成立）は特徴的といえる。

（２）自分が０の値を持つ場合（イベント不発生）には、他の正規化ログにおいて１の値を持つ（イベント発生）ログを集計し、その集計値を０から引いた値をログ特徴として設定する。この場合、ログ特徴の絶対値が高ければ、当該イベントの不発生（不成立）は特徴的といえる。

たとえば、正規化ログＡのイベントｂについて、その正規化値は１であって、且つこの場合、該イベントの正規化値が０であるものは正規化ログＢとＤであるから、正規化ログＡのイベントｂのログ特徴を示す値は２となる。また、正規化ログＡのイベントｆについて、その正規化値は０であって、且つこの場合、該イベントの正規化値が１であるものは正規化ログＢの一つだけであるから、正規化ログＡのイベントｆのログ特徴の値は０から１を引いて－１となる。

正規化ログＡ～Ｄそれぞれのログ特徴値（配列）Ａ～Ｄは、次の通りとなる。なお、図３はこれをグラフ化したものである。

特徴値Ａ：（　２，０，－１，－１，－１，０）

特徴値Ｂ：（－２，０，　３，－１，－１，０）

特徴値Ｃ：（　２，０，－１，　３，　３，０）

特徴値Ｄ：（－２，０，－１，－１，－１，０）

次にステップＳ５において、類似度算出部８は、ある部分ログと他の部分ログとの全ての組み合わせ（ＡＢ，ＡＣ，ＡＤ，ＢＣ，ＢＤ，ＣＤ）において、部分ログ同士の類似度を特徴値Ａ～Ｄに基づく内積演算により算出する。内積演算の値が大きいほど、その部分ログ同士は他の組よりも類似度が高いものと判定する。例えば部分ログＡとＢとの比較（組み合わせ）において、これら部分ログそれぞれの特徴値ＡとＢとの内積演算は次の通りとなる。

ＡＢ＝２＊（－２）＋０＊０＋（－１）＊３＋（－１）＊（－１）
＋（－１）＊（－１）＋０＊０
＝－５

上記全ての組み合わせについて演算を行なった結果は次の通りとなる。

ＡＢ＝－５

ＡＣ＝－１

ＡＤ＝－１

ＢＣ＝－１３

ＢＤ＝　３

ＣＤ＝－９

ここで、ステップＳ６において、デバッグ作業者１２が部分ログ指定部１１を介して比較対象の元となる部分ログを指定すると、部分ログ選択部９は、該指定部分ログと類似度の高い部分ログを、類似度算出部８により算出された類似度に基づき選択する。例えばデバッグ作業者１２が、部分ログ指定部１１を介して部分ログＤ指定したとすると、ステップＳ７においてログ選択部９は、部分ログＤに最も類似する部分ログとして部分ログＢを選択する。

【００２３】

そしてステップＳ８において、ログ表示部９は、部分ログＢと部分ログＤとを、好ましくは両者の相違部分を強調して表示する。なお、部分ログＢに類似する部分ログを、その類似度順に並べ替えて表示してもよい。

【００２４】

.なお、部分ログ指定部１１により特に部分ログの指定を行なわない構成を採る場合には、上記全ての組み合わせについてその類似度を表示すればよい、この場合、類似度順に並べ替えて表示しても良い。

【００２５】

以上説明したように、第１実施形態に係るログ比較デバッグ支援装置は、正規化ログ同士を単純に比較するのではなく、主ログの構成要素イベント（本実施形態では「ｂｃｆｄｅｇ」）それぞれの発生及び不発生の特徴度合いを示す特徴値Ａ～Ｄを算出し、該特徴値の内積演算に基づいているので、稀に発生する（もしくは稀にしか欠けない）イベントを共通に含むことが重視された類似判定を行なうことができる。

【００２６】

部分ログＡとＤ，ＢとＤを比べると、いずれも１イベントの相違しかないが、ＡとＤの違いである１番目のイベント「ｂ」と、ＢとＤの違いである３番目のイベント「ｆ」とを比べると、１番目のイベントがＢとＤで共通であることの方が、３番目のイベントがＡとＤで共通であることよりも稀（特徴的）である。つまり、発生又は不発生が特徴的なイベントを共通に含むことが重視された類似度判定を行なうことができる。

【００２７】

したがって、動作ログＬのデータ量が大きくなってデバッグ調査の範囲が拡大するような場合であっても、着目箇所に関する基準の部分ログを指定するだけで、これに類似しログ比較に有益な部分ログを容易に得ることができ、プログラム動作理解を含めたログベースのデバッグ作業の効率向上に寄与する。

【００２８】

（第２実施形態）

次に、本発明の第２実施形態を説明する。

図４は、本発明の第２実施形態に係るログ比較デバッグ支援装置の概略構成を示すブロック図である。当該第２実施形態は、対象プログラムのソースコードを基準としたログ比較を行なうデバッグ支援装置に関するものであり、第１実施形

態よりも動作ログＬの構成内容が現実的である。

【００２９】

また、第２実施形態の装置では、主ログ生成部７がデバッグ対象のプログラムのソースコード７を入力し、該ソースコードを展開処理して主ログを生成する点に特徴がある。他の構成要素については第１実施形態のものと同様である。

【００３０】

当該第２実施形態の大まかな処理の流れは、第１実施形態と同様である。以下、図５に示すプログラム７の実行時のログとして、動作ログＡ～Ｈからなるものが得られたとして説明する。図６に動作ログＤを、図７に動作ログＥを示し、他の動作ログについては図示省略する。

【００３１】

先ず、デバッグ作業者（ユーザ）１２は、ログ比較条件として、図５に示されているソースプログラム中に記述されているｍａｉｎ関数を対象関数として指定する。また、ソースプログラムの展開パラメータとして、該ｍａｉｎ関数内における関数の再帰呼び出しの記述を４段階まで展開し、及びループ構文を３回まで展開する旨を表すパラメータを指定する。また、ｍａｉｎ関数内における代入文、コメント、および空白行の記述は無視（主ログにおいて削除）する旨の指定も行なう。

【００３２】

これにより主ログ生成部４は、対象プログラムのソースコード７を入力し、該ソースコードにおけるｍａｉｎ関数の記述部分を切り出し、条件指定部１により指定された展開パラメータに従って展開し、その展開結果を主ログとして生成する。これにより生成された主ログを図８に示す。図８から分かるように、再帰呼び出しに係る関数「makeValueString(...)」は４段階にわたって展開記述されている。

【００３３】

部分ログ生成部２は条件指定部１に与えられた展開パラメータに従い、動作ログＡ～Ｈのそれぞれから、指定関数としてのｍａｉｎ関数につき、その実行開始から終了までを部分ログとして切り出す。なお、図６に示された動作ログＤ、及

び図７に示された動作ログＥは、共にｍａｉｎ関数実行部分に対応するログであるから、そのまま部分ログとなる。また、図示していないが、動作ログＤ及びＥ以外の動作ログ（Ａ，Ｂ，Ｃ，Ｆ，Ｇ，Ｈ）については、それぞれ１つのｍａｉｎ関数の実行ログを含むものと仮定して説明する（これらの正規化ログについては図１０に示す）。

【００３４】

正規化ログ生成部５は、部分ログ生成部２から得られた部分ログと、主ログ生成部４から得られた主ログとを対応させ、主ログの構成要素イベントが当該部分ログに存在する場合には１、存在しない場合には０が設定されたビット列を生成する。これを正規化ログとする。動作ログＤの部分ログのプログラム記述に対応した正規化の内容を図９に示す。また、当該第２実施形態における部分ログＡ～Ｈそれぞれの正規化ログの一覧を図１０に示す。

【００３５】

ログ特徴値算出部５は、正規化ログ生成部５により生成された正規化ログに基づき、第１実施形態で行ったものと同様の演算を行って特徴値（配列）Ａ～Ｈを算出する。特徴値Ａ～Ｈの一覧を図１１に示す。

【００３６】

ログ類似度算出部８は、ログ特徴値算出部５により算出された特徴値Ａ～Ｈに基づき、第１実施形態で行なったものと同様の内積処理を行って類似度を算出する。部分ログＡ～Ｈの全ての組み合わせについて算出した類似度の一覧を図１２に示す。

【００３７】

ここで、部分ログ指定部１１により、デバッグ作業者１２が比較の基準となるログとして部分ログＤを選択したとする。ログ選択部９は、ログ類似度算出部８により算出された類似度に基づき、部分ログ指定部１１により指定された部分ログＤに最も類似する部分ログとして、類似度が最大値「４７０」となる部分ログＨを選択する。

【００３８】

第２実施形態によれば、第１実施形態と同様に、主ログの構成要素イベント（

本実施形態では関数呼び出し、ループ、スイッチなどの構文記述である）それぞれの発生及び不発生の特徴度合いを示す特徴値Ａ～Ｈを算出し、該特徴値の内積演算に基づいているので、稀に発生する（もしくは稀にしか欠けない）イベントを共通に含むことが重視された類似判定を行なうことができる。

【００３９】

したがって、比較的多数の動作ログＡ～Ｈによりデータ量が大きくなってデバッグ調査の範囲が拡大するような場合であっても、着目箇所に関する基準の部分ログを指定するだけで、これに類似しログ比較に有益な部分ログを容易に得ることができ、プログラム動作理解を含めたログベースのデバッグ作業の効率向上に寄与する。

【００４０】

なお、本発明は上述した実施形態に限定されず種々変形して実施可能である。例えば、部分ログの特徴値同士の演算は内積演算によるものとして説明したが、他のアルゴリズムに基づく演算を行なってもよい。

【００４１】

また、上述した実施形態には種々の段階の発明が含まれており、開示される複数の構成要件における適宜な組み合わせにより種々の発明が抽出され得る。例えば、実施形態に示される全構成要件から幾つかの構成要件が削除される場合である。より具体的には、条件指定部１、部分ログ生成部２、および主ログ生成部４のうちの一つ又は全部に相当する要件が削除される場合である。つまり、部分ログを自動生成しないで直接的に入力したり、部分ログの結合やソースプログラムを参照した主ログの自動生成を行なわず事前に用意された主ログを直接的に入力するように構成される。この場合、適切な部分ログ及び主ログを準備する手数がかかるものの、ログ比較に有益な情報が得られるという発明の作用効果を得ることができる。

【００４２】

【発明の効果】

以上説明したように、本発明によれば、ログ比較に有益な部分ログを得ることができ、ログベースのデバッグ作業の効率向上に寄与するログ比較デバッグ支援

装置を提供できる。

【図面の簡単な説明】

【図１】

本発明の第１実施形態に係るログ比較デバッグ支援装置の概略構成を示すブロック図

【図２】

実施形態に係るログ比較デバッグ支援装置の大まかな処理の流れを示す図

【図３】

第１実施形態に係る正規化ログＡ～Ｄそれぞれのログ特徴値（配列）Ａ～Ｄを示すグラフ

【図４】

本発明の第２実施形態に係るログ比較デバッグ支援装置の概略構成を示すブロック図

【図５】

第２実施形態に係る対象プログラムのソースコードを示す図

【図６】

第２実施形態に係る対象プログラムの実行時のログの一例を示す図

【図７】

第２実施形態に係る対象プログラムの実行時のログの他の例を示す図

【図８】

第２実施形態に係る対象（ソース）プログラムに基づき生成された主ログを示す図

【図９】

第２実施形態に係る動作ログＤの部分ログのプログラム記述に対応した正規化の内容を示す図

【図１０】

第２実施形態に係る部分ログＡ～Ｈそれぞれの正規化ログの一覧を示す図

【図１１】

第２実施形態に係る部分ログＡ～Ｈそれぞれの特徴値の一覧を示す図

【図１２】

.第２実施形態に係る部分ログＡ～Ｈの全ての組み合わせについて算出した類似度の一覧を示す図

【符号の説明】

１…条件指定部

２…部分ログ生成部

４…主ログ生成部

５…正規化ログ生成部

６…ログ特徴値算出部

７…対象プログラム（のソースコード）

８…ログ類似度算出部

９…ログ選択部

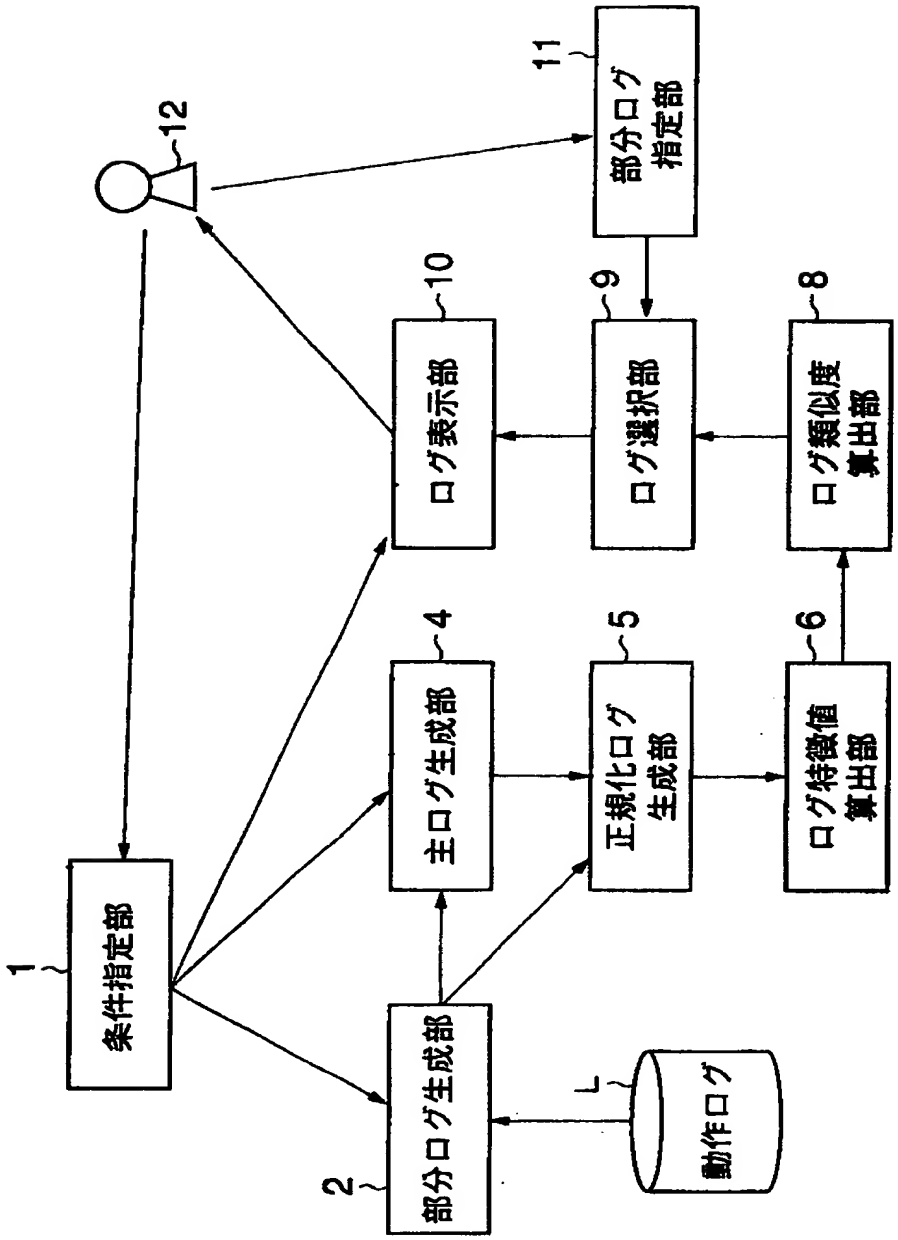
１０…ログ表示部

１１…部分ログ指定部

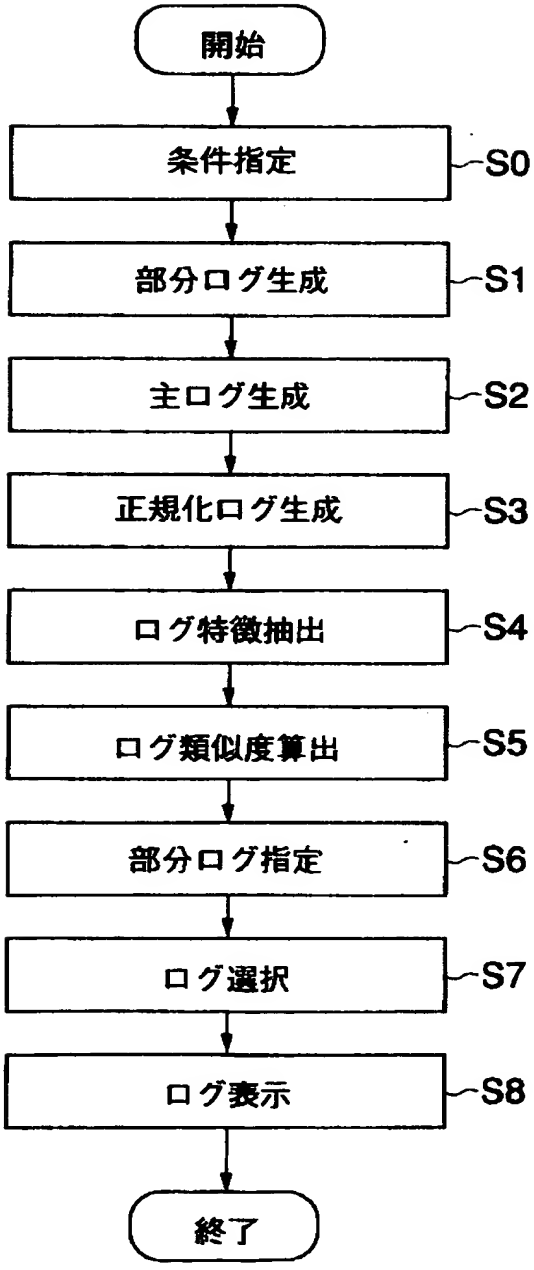
１２…デバッグ作業者（ユーザ）

【書類名】　　図面

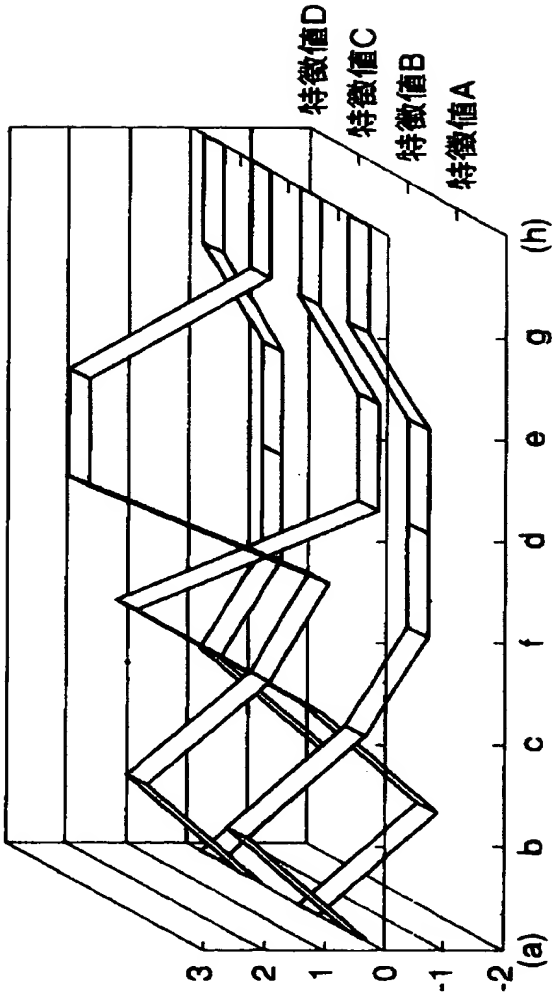
【図１】


条件指定部
1
部分ログ生成部
2
主ログ生成部
4
正規化ログ生成部
5
ログ特徴値算出部
6
ログ類似度算出部
8
ログ選択部
9
ログ表示部
10
部分ログ指定部
11
12
動作ログ
L

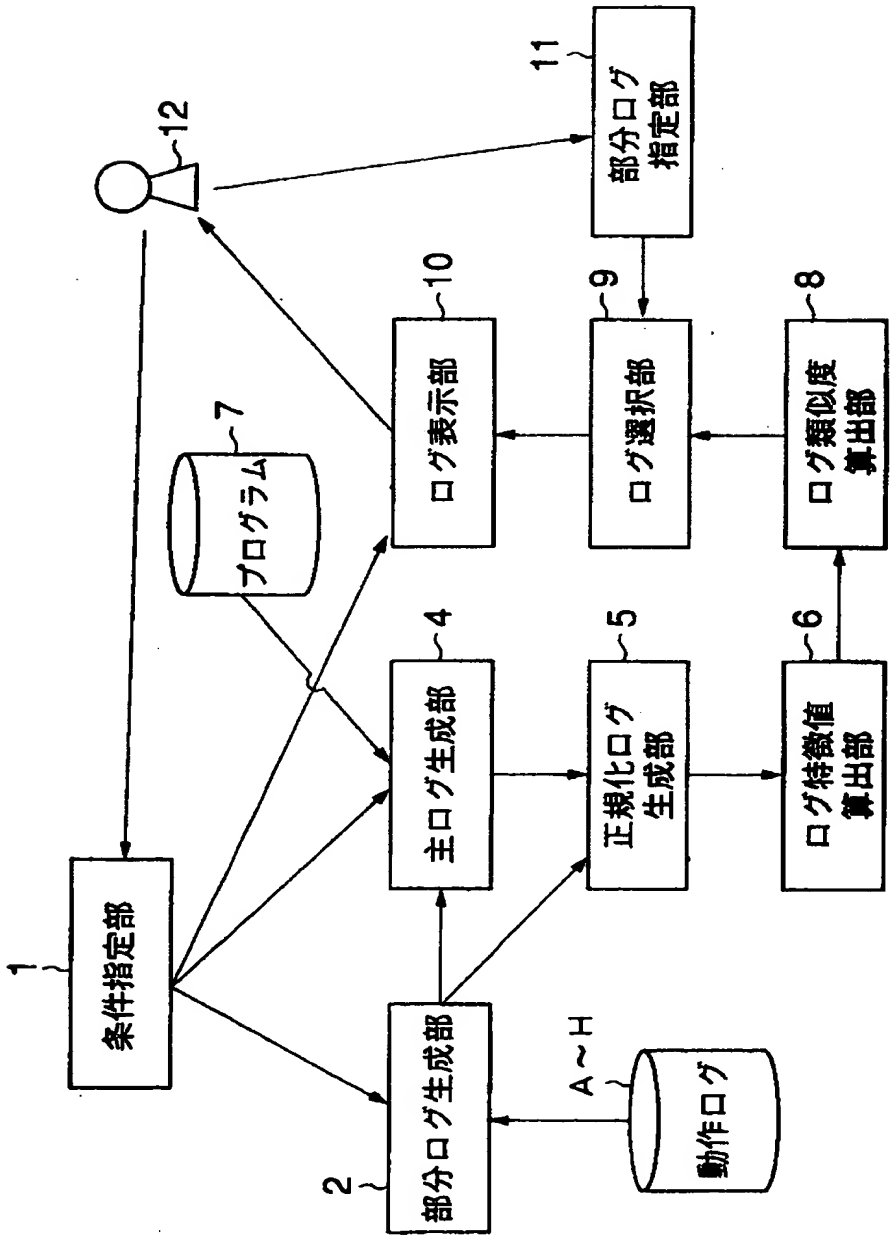
【図２】


開始
条件指定
S0
部分ログ生成
S1
主ログ生成
S2
正規化ログ生成
S3
ログ特徴抽出
S4
ログ類似度算出
S5
部分ログ指定
S6
ログ選択
S7
ログ表示
S8
終了

【図３】



【図４】


1
条件指定部
2
部分ログ生成部
A～H
動作ログ
4
主ログ生成部
5
正規化ログ生成部
6
ログ特徴値算出部
7
プログラム
8
ログ類似度算出部
9
ログ選択部
10
ログ表示部
11
部分ログ指定部
12

【図５】

```
#include<stdio.h>
#define MAX_STR_NUM 10

// 関数の宣言
void PrintError();
void reverseString(char* string);
void makeValueString(int value,char* str);

// main 関数
int main(int argc,char** argv){
     char string[MAX_STR_NUM];
     // 引数の数を3進数表記の文字列に変換する :文字の順序が逆)
     makeValueString(argc,string);
     // 文字の順序を逆に戻す
     reverseString(string);
     // 結果表示する
     printf(" %d to %s\n",argc,string);
 }

// 各関数の定義
void makeValueString(int value,char* str);
 {
    // 再帰関数の再帰終了条件
    if(value<=0)
       {
         str[0]='\0';
         return;
       }
    makeValueString( value/3,str+1);
    switch(value%3)
       {
       case 0:
         str[0]='\0'; // '0'の間違い
         break;
       case 1:
         str[0]='1';
         break;
       case 2:
         str[0]='2';
         break;
       default:
         break;
       }
 }

void PrintError()
 {
   printf("error\n");
 }

void reverseString(char* string)
 {
   char tmp  char
   int n;
   int i;
   n=strlen(string);
   if(n==0)
      {
        PrintError(); // エラー処理
      }
   else
      {
       // 文字列の順序を逆にする
       for (i=0;i<(n/2);i++)
          {
            tmp_char=string[i];
            string[i]=string[n-1-i];
            string[n-1-i]=tmp_char;
          }
      }
}
```

【図６】

```
main(12,0x10000)
  {
    makeValueString(12,0x20000)
      {
        if(value<=0)
        makeValueString(4,0x20001)
          {
            if(value<=0)
            makeValueString(1,0x20002)
              {
                if(value<=0)
                makeValueString(0,0x20003)
                  {
                    if(value<=0)
                      {
                      }
                  }
                switch(value%3)
                  {
                  case 1:
                  }
              }
            switch(value%3)
              {
              case 1:
              }
          }
        switch(value%3)
          {
          case 0:
          }
      }
    reverseString(0x20000)
      {
        n=strlen(0x20000);
        if(n==0)
          {
            PrintError()
              {
                printf(" error\n").
              }
          }
      }
    printf(" %d to %s\n",12,0x20000);
  }
```

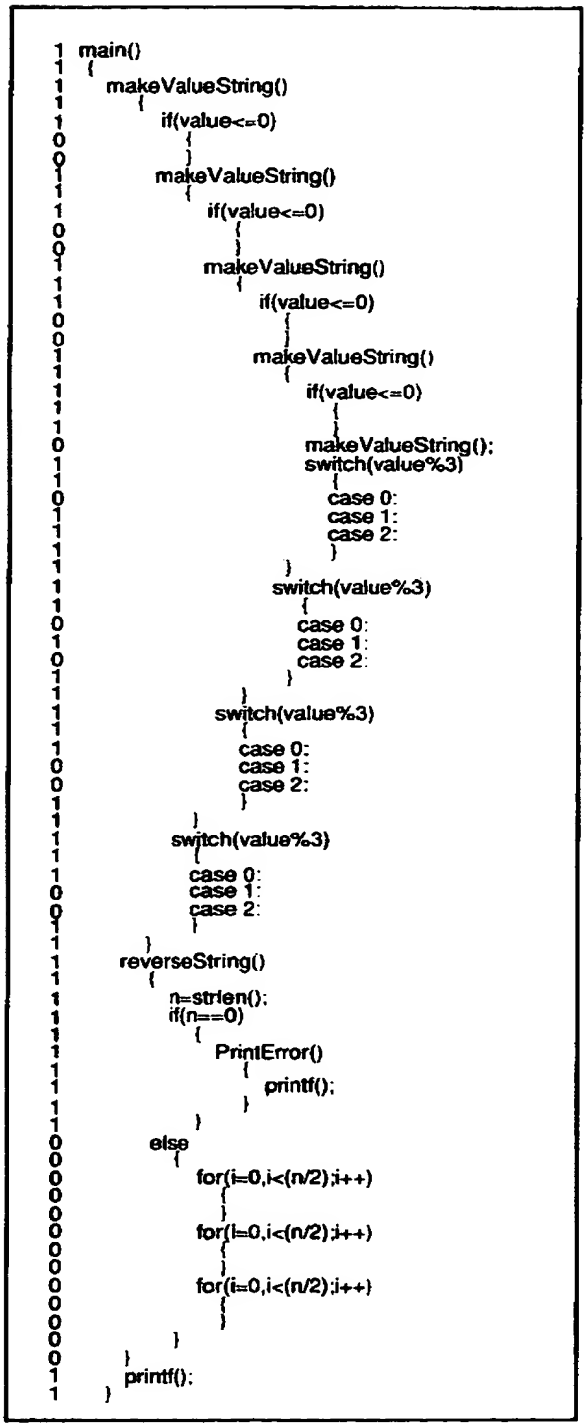
【図７】

```
main(13,0x10000)
  {
    makeValueString(13,0x20000)
      {
        if(value<=0)
        makeValueString(4,0x20001)
          {
            if(value<=0)
            makeValueString(1,0x20002)
              {
                if(value<=0)
                makeValueString(0,0x20003)
                  {
                    if(value<=0)
                      {
                      }
                  }
                switch(value%3)
                  {
                  case 1:
                  }
              }
            switch(value%3)
              {
              case 1:
              }
          }
        switch(value%3)
          {
          case 1:
          }
      }
    reverseString(0x20000)
      {
        n=strlen(0x20000);
        if(n==0)
        else
          {
            for(i=0,i<(n/2);i++)
              {
              }
          }
      }
    printf(" %d to %s\n",13,0x20000);
  }
```

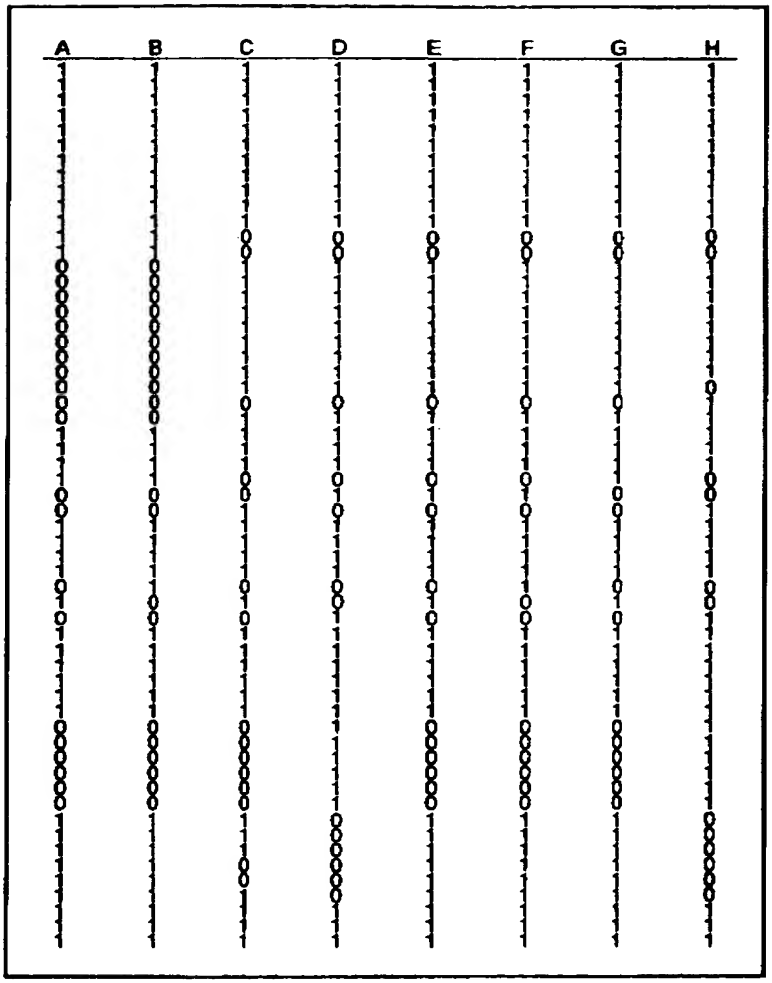
【図８】

```
int main(int argc,char** argv)
  {
    makeValueString(int value,char* str)
      {
        if(value<=0)
          {
          }
        makeValueString(int value,char* str)
          {
            if(value<=0)
              {
              }
            makeValueString(int value,char* str)
              {
                if(value<=0)
                  {
                  }
                makeValueString(int value,char* str)
                  {
                    if(value<=0)
                      {
                      }
                    makeValueString(value/3,str+1);
                    switch(value%3)
                      {
                        case 0:
                        case 1:
                        case 2:
                        }
                  }
                switch(value%3)
                  {
                    case 0:
                    case 1:
                    case 2:
                    }
              }
            switch(value%3)
              {
                case 0:
                case 1:
                case 2:
                }
          }
      }
    reverseString(char* string)
      {
        n=strlen(string);
        if(n==0)
          {
            PrintError()
              {
                printf(" error\n");
              }
          }
        else
          {
            for(i=0,i<(n/2);i++)
              {
              }
            for(i=0,i<(n/2);i++)
              {
              }
            for(i=0,i<(n/2);i++)
              {
              }
          }
      }
    printf(" %d to %s\n",argc,string);
  }
```

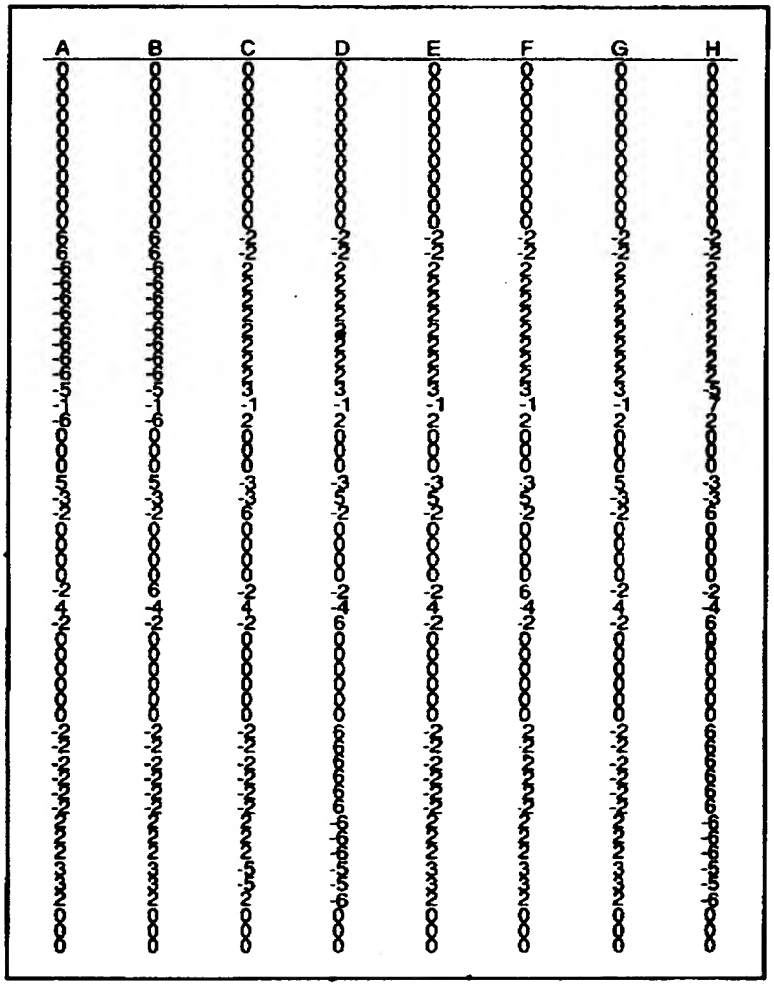
【図９】


main()
{
makeValueString()
{
if(value<=0)
{
}
makeValueString()
{
if(value<=0)
{
}
makeValueString()
{
if(value<=0)
{
}
makeValueString()
{
if(value<=0)
{
}
makeValueString();
switch(value%3)
{
case 0:
case 1:
case 2:
}
}
switch(value%3)
{
case 0:
case 1:
case 2:
}
}
switch(value%3)
{
case 0:
case 1:
case 2:
}
}
switch(value%3)
{
case 0:
case 1:
case 2:
}
}
reverseString()
{
n=strlen();
if(n==0)
{
PrintError()
{
printf();
}
}
else
{
for(i=0,i<(n/2);i++)
{
}
for(i=0,i<(n/2);i++)
{
}
for(i=0,i<(n/2);i++)
{
}
}
}
printf();
}

【図１０】



【図１１】



【図１２】

	A	B	C	D	E	F	G	H
A		494	-130	-346	-90	-138	-26	-308
B			-178	-330	-138	-58	-74	-290
C				-58	-70	22	-70	-18
D					-82	-66	-146	470
E						126	110	-170
F							62	-154
G								-170
H								

【書類名】　　要約書

【要約】

【課題】ログ比較に有益な部分ログを得ることができ、ログベースのデバッグ作業の効率向上に寄与するログ比較デバッグ支援装置を提供すること

【解決手段】対象プログラムの実行によって発生した一連のイベントが記録されたログを入力し、該入力ログから複数の部分ログを生成する。正規化規準としての主ログに基づきこれら部分ログを正規化し、それぞれの正規化ログについて、イベントの発生及び不発生の特徴度合いを示す特徴値を他の部分ログの正規化ログに基づいて算出する。所定の部分ログと他の部分ログとの組み合わせにおいて、特徴値に基づく演算により該部分ログ同士の類似度を算出する。例えば、最も類似度の値が高い部分ログの組み合わせを、相違部分を強調して表示出力する。

【選択図】　　図１

出願人履歴情報

識別番号　〔０００００３０７８〕

１．変更年月日　１９９０年　８月２２日
［変更理由］　新規登録
住　所　神奈川県川崎市幸区堀川町７２番地
氏　名　株式会社東芝